

The Official **AMOS**

CLUB NEWSLETTER Vol 2 Issue 4

It returns ! It's been a long time since you last saw an AMOS Newsletter ! Most of the delay between issues was caused by AMOS Professional being released just as Volume 2, Issue 3 went out. Now normally, a new product going out wouldn't cause much of a problem. However, AMOS Pro was pretty badly bugged and unreliable, causing a lot of new AMOS owners to phone Europress on the helpline number that they promised would give unlimited technical help on their registration card. Unfortunately, there isn't a great deal of deep AMOS technical knowledge going around amongst the helpline people at Europress, so most of the callers were told to phone the AMOS Helpline here. The end result was that I was bombarded with disgruntled new AMOS Pro users, most of whom were not AMOS Club members (or they would have phoned here first anyway), blocking up the helpline over Christmas and the New Year and delaying my newsletter and program writing. I am now catching up on my work (The Newsletter, Issue 4 of "The Disk", Jetstrike, Sprite X 3.0, The Shuffle Extension Manual and disk, TOME Goodies Disk 2, TOME 4.5 amongst others) and my apologies for the delay. As

mentioned before, your subscriptions are for 6 issues of the newsletter rather than a fixed time, so they will only run out after issue 6 is released.

Don't let this delay worry you about using the helpline though, as I can cope quite easily with AMOS Club members phoning up with technical problems. In fact I rather enjoy fixing them (it also helps me find out what you want in the way of new bits for AMOS).

New Toys

There are 2 new extensions being released soon (or already). Firstly, there's Nigel Critten's Sticks extension, which gives you access to just about every Joystick and Mouse input going. It handles "Sega" type multi button joysticks (2 and 4 buttons), both Mouse

In This Issue:

Using Printers in AMOS	4
Fixing the WAIT KEY bug	6
System Calling	7
Converting to Easy AMOS	9
Hardware Scrolling with AMOS TOME	11
Using Float Bob	13
10 Liners (10 Line Splitter, XoTron & Treefender	14
Foreign Affair (writing multi- lingual programs)	17
Disk Lister & File Finder	18

printed manual (around 24 pages) and will cost around £15. Like Ctext 2 it will only be available to AMOS Club members.

From the response we've been getting from TOME Series IV users, it seems to be rather popular. However, while working on Jetstrike (out in September 93) I've developed a higher speed version of TOME, which goes directly to the Blitter chip and works out around 2.5 to 3 times as fast ! However this particular version can only be used in Jetstrike, as it is directly set up for this game, but I will be using the same high speed routines in TOME 4.5, which TOME Series IV users will be able to get and use as an optional faster version. It has built in Hardware scrolling routines and I'm working on the word per tile format map, so that you can use up to 65536 tiles (as long as you have the memory). Experimental at the moment, but when released it will be available as a P.D. Upgrade disk to all TOME IV users. Keep in touch on the helpline for the latest information.

Just about all of you will have heard of this game in development at some point, as I've been working on it since AMOS was first written (in fact, a lot of AMOS extension commands are direct results of Jetstrike work). Well, you'll (hopefully) be glad to hear that it is nearly finished (honest!) and will be released in September 93.

AMOS commands) and is FUN ! Go down to your Computer shop and tell the shop keeper to order you a copy as soon as it's available ! (Now if that isn't a shameless plug, nothing is !)

GamesMaster Comp Winner

The game itself is in the Magic Forest/ Mario etc. mould and plays quite well. It is available from Brian Bell at 8 Magnolia Park, Twinbrook, Dunmurry, Belfast. BT17 0DS. Charlie Chimp costs £5.00, mention that you are an AMOS Club member (with your membership number) and Brian will throw in Charlie Chimp II free. Also available is the sourcecode (£8.00 for AMOS Club members).

A hello to the latest AMOS Group to form, the Danish AMOS User group, run by Tom Poulsen. They have a bi-monthly disk based magazine with articles on programming, members own programs and reviews of selected AMOS P.D Titles.

**The Danish AMOS User Group,
Stenmullen 28,
2640 Hedehusene,
Denmark**

Hopefully, we'll be seeing Tom and some of the other Danish AMOS users at this year's OZ Happening (September 11th & 12th), talking of which....

The OZ Happening 92 !

Programmers, Graphic artists, musicians and designers were invited from all over Europe, including a great deal of AMOS Users. All the European AMOS Clubs were there, and programming tricks were passed around almost as freely as the beer !

As it was a meet and greet type show, rather than a "sales" event, all the members of the public that came to the show were able to discuss their programs with the people who actually wrote them, and the programmers got a lot of useful feedback from the users. In all, an excellent event, which I will be attending again this year. With access to Europe a lot easier now, I hope to see a lot of you there !

By the way, here's the world's smallest game, when printed onto a sticker, it goes on the back of a postage stamp!

```

Dim S(50) L=1:Col=7:FF0 Cur
On Cbo0 Value <10 and >0 Paper
If or A=1 to L S(A)=Print(3):C=0
Home PenS(A)=4:Pr S(A)=1
Zoom=0,9:24,81 to 80:50:24,0:10,
Play15,S(A)^4+35:20:50:24,0
B=1 While B=End and D=0 A$=""
While (A$<"1") or (A$>"4") A$=""
While Word In A$ (A$)=S(B):
Inc B Play15, S(B)^4+35:20:Else
D=1:End If Word:Inc L Word
If D Print "You Scored 1, Else
Print A$ Parhed 10: End If

```

Using the printer in AMOS

Ever since the non existant LLIST command was mentioned in the AMOS Manual everyone with a printer has had some sort of problem in using it from AMOS.

The printer access commands in AMOS are slightly bugged, or at least not user friendly, and due to the way printers are accessed from the Amiga, there is a lot of misunderstanding over how to control the printer.

Lprint Vs. Open Port 1,"PRT:"

When you first use an **Lprint** command, it automatically opens the printer device, doing the same as an **Open Port 1,"PRT:"** command.

However, once opened with **Lprint**, the device cannot be closed again within the program. This can lead to some problems (usually lockouts). So rather than using the **Lprint** command, you should use the **Open Port** command and then use **Print #1**, rather than **Lprint** to send data to the printer. This has two major advantages in that

- a) you can Close the device again later and
- b) that you can quite easily re-direct your printing to a disk file by changing the **Open Port 1,"PRT:"** to **Open Out 1,"Filename"**.

Using ANSI Codes

The Amiga is a clever little beastie ! Knowing that just about every printer has its own set of custom control codes to access functions such as Underline On/Off, setting the nationality of the font and so on, the

Amiga uses the ANSI printer code system so that your program running on the Amiga talks to the Amiga using the standard ANSI codes, and the printer driver in the Amiga converts these codes to the correct ones for your printer.

This however has caught a lot of people out, as they have tried to send the custom character codes for their printer to the printer driver. For instance, to switch on the Underline on an Epsom printer, you would do: `Print #1,Chr$(27);"-";Chr$(1)` However, your Amiga won't recognise this and will ignore it, whereas the ANSI code (Which your Amiga will understand) is:

```
Print #1,Chr$(27);"[4m"
```

There are all sorts of fun codes in the ANSI specification, and fortunately someone has put them all in a book. The good news is that the book is your Amiga User's manual (you know the one you got in the box with your machine and put away somewhere. I found mine down the back of my desk, hidden behind a heating pipe !).

In my copy (A2000/1500 English) it is in Appendix D-3/4 although with the wonderful way the big C work, it could be anywhere in your manuals, have a look !

The only problem with this system is that you must have the correct printer driver installed on your boot disk for your printer (as with the Dump extension, as you'll see next).

The Dump Extension

Printer dumps were something that AMOS didn't handle very well at first (i.e it couldn't do it !), although thanks to a little routine called the Dump extension written by Alex J

Grant in Australia, and then updated by Francois, it is now easy to do graphic printer dumps.... Ah well, it would have been, except:

- 1) The extension needs the WB2 Printer Device (now on most up to date AMOS disks)
- 2) Francois Modified the commands, but he didn't modify the docs.
- 3) You must have the correct printer driver installed on your boot disk for it to work.
- 4) Although the official version is supposed to be extension number 10, versions floating around have been on extensions 15,20 and 25 !

To make life a little easier, I've compiled an official Dump Extension PD disk, with the correctly modified documentation, the correct Printer Device file and the extension assembled as extension number 10. Contact the AMOS P.D Library for a copy (they can also supply disks of printer drivers). The full dump extension is also on issue 4 of The Disk.

Printer Drivers

As mentioned above, you need the correct printer driver installed on your BOOT disk for your printer routines to work. If you work most of your software from hard disk, the correct driver just has to be selected from preferences and stored on the hard disk for it to be correct for all your programs. If however you boot your AMOS from floppy disk, you need to copy the printer driver onto your AMOS disk (a backup of course), so that it will work. Unfortunately, this means that you will need to boot your Workbench disk first, select the printer preferences and from there select the printer driver you want to

use. When you have set up the printer preferences ready for your printer, save them to your Workbench disk (which of course is a backup version) and select the **CopyPrefs** icon from inside the **Prefs** directory. This will copy the preferences from your Workbench disk onto the disk in **DF0:**. This is obviously easier if you swap your WB disk into **DF1:**, or simply copy the file **DEVS:System-Configuration** into the DEVS directory of your AMOS Disk, along with the **Printer.Device** file.

You now need to create a directory called **PRINTERS** in the DEVS directory of your AMOS Disk: **Mkdir "Devs/Printers"** will do this, as will **SID**, **OPUS**, **CLIMate** or any other file utility program.

You then copy your printer driver across from the Devs/Printers directory of your Workbench disk to Devs/Printers of your AMOS boot disk.

Eventually, you should end up with a bootable version of AMOS that can correctly access your printer !

Getting the correct printer driver for your shiny new printer can be a problem, as it is usually a while before a driver will appear in P.D, although most of the major printer manufacturers are very good at getting a correct driver out as soon as possible. If you cannot find the printer driver you want on P.D, then try contacting the manufacturer, so far we've found that both Canon and Star have both given excellent support to Amiga users who haven't been able to get drivers for their printers, and other manufacturers

have been known to support their printers just as well.

And Finally!

By the way, **LLIST** doesn't exist, it never has, and it never will exist in AMOS! It was a command in STOS to list the program to the printer, and accidentally got listed in the early AMOS manuals. Because it was line number based, the command had no use in AMOS, so the method of listing an AMOS program was made easier. i.e. you mark the section you want to list as a block (Ctrl-A marks the whole program) and then select Block Print from the Blocks Menu (Ctrl-F10).

For those of you who have an unofficial version of the Dump extension (on any other extension number than 10), the following program will convert it to using extension 10. Type in and run the following listing, load up your Config1_3.AMOS program, load up the default configuration, and move the Dump.Lib entry from whatever extension it was on to extension 10. Once this is done, reboot AMOS, and dump should be re-installed as extension 10.

```
1) Screen Open 0,640,200,4,
Hires : Curs Off : Flash Off
: Cls 0 : Paper 0 : Pen 1 :
Colour 1,$ FFF [86]
2) DN$=":AMOS_System/Dump.
Lib" [41]
3) Open In 1,DN$ [33]
4) L=Lof(1) [12]
5) Close 1 [71]
6) If L<>1432 [110]
7) Print "You do not have a
normal version of the Dump
Extension !" [58]
8) Else [62]
```

```
9) Reserve As Work 5,L :
Bload DN$,Start(5) [38]
10) Loke Start(5)+$A4,
$1887009 [15]
11) Loke Start(5)+$130,
$188D4FC [40]
12) Loke Start(5)+$164,
$246D0188 [4]
13) Loke Start(5)+$1F0,
$246D0188 [112]
14) Loke Start(5)+$488 ,
$246D0188 [3]
15) Bsave DN$,Start(5) To
Start(5)+L [61]
16) Print "You now have an
official Dump extension, load
config1_3.amos and install it
as extension 10" [127]
17) End If [64]
As Usual, don't enter the line
number before the bracket, or the
checksum in the square brackets.
```



Fixing the WAIT KEY Bug!

Since AMOS got upgraded to 1.34, there has been a slight bug with the Wait Key command, in that when you compile the command it will crash the Amiga if you hit the mouse button while it is waiting for a key. This is because Francois left a jump to his Debugging routines in the Wait Key command (he normally removes it before releasing it), and with the Debug routines only being on his Amiga, it's usually red box time for the rest of us!

This patch program hunts down the Wait Key command in your AMOS.Lib file (this is the Compiler version of all the AMOS commands),

and eliminate the call to the Debug routine. As with all modifications to AMOS, only do this with a backup of AMOS.

```
1) F$="amos_system/AMOS.lib"
[66]
2) Open In 1,F$ [96]
3) L=Lof(1) [12]
4) Close 1 [71]
5) Reserve As Work 5,L [38]
6) Bload F$,Start(5) [95]
7) S$=Space$(28) [32]
8) Loke Varptr(S$),$FE3100CF
[69]
9) Loke Varptr(S$)+4,$8390006
[112]
10) Loke Varptr(S$)+8,$BFE001
[34]
11) Loke Varptr(S$)+12,
$660E7200 [4]
12) Loke Varptr(S$)+16,
$206DFFF8 [126]
13) Loke Varptr(S$)+20,
$4EA80120 [126]
14) Loke Varptr(S$)+24,
$4AFC4E75 [114]
15) H=Hunt(Start(5) To
Start(5)+L,S$) [85]
16) For A=0 To 27 Step 2 [29]
17) Doke H+A,$4E71 [119]
18) Next A [97]
19) Bsave F$,Start(5) To
Start(5)+L [81]
```

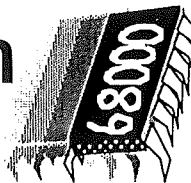
THE AMOS HELPLINE



0271 23544

As an AMOS Club member, you can call us when you need technical help. Remember to quote your membership number.

System Calling



By Aaron Fothergill

As we've seen from previous DOS Calling articles, you can greatly expand the power of AMOS by using the Amiga system routines through the DOSCALL, EXECALL, INTCALL and GFXCALL functions. In this article, I'm going to show you a few more of the functions that you can use.

Protecting Files

The DOS Command **SetProtection** allows you to set the 4 protection bits of any Amiga Dos file or directory, allowing you to make the file non deletable, non executable, non overwritable and even non readable!

The command offset is -186, it requires the address of the filename in D1 and the mask in D2. The bits to set in the mask are:

0= File Not Deletable,
1= Not executable,
2=Not Overwritable,
3=Not Readable.

This means that you can make a file non deletable and non overwritable with %101 (bits 0 and 2 set) or non readable with %1000 (Bit 3 set).

Here's an example that protects a file from being deleted or overwritten

```
F$="My_Program.AMOS"
```

```
Dreg(1)=Varptr(F$) : Rem
point D1 to string
```

```
Dreg(2)=%101 : Rem no delete,
no overwrite
```

```
Z=Doscall(-186) : Rem Do it
```

If you save this as **My_Program.AMOS** and run it, you

will find that you cannot delete the program from disk, or overwrite it (try hitting save again !). In fact, the only way to get rid of it is to change the third line to:

```
Dreg(2)=0
```

and run it again, this will remove the protection from the file, so that you can get rid of it.

The **Non Readable** flag isn't as silly as it sounds. If you want one of your sprite banks to only be able to be read from within your game for instance, you can set it to non readable, but within your game make it readable again, load it, and then set it to non-readable !

It isn't a perfect form of protection, as anyone who knows about this system can easily get around it, and it does involve writing to disk (not a great idea on a commercial game), but it will allow you to slow down the spread of your graphics !

O Forbid & Permit

Much has been said by programmers about these functions, but very few seem to know what they are !

In fact, they are rather handy routines that allow you to speed up your program by switching off multi-tasking and then switching it back on again. Like the Hide and Show commands however, they stack up. So if you did 2 Forbids and a Permit, Multi tasking would still be off until you did another Permit. Watch out for that, it can catch you out.

The only drawback of using the Forbid function, is that it locks out the keyboard and mouse (as these are read by a separate task).

However, the **=Key Scan** function, in the latest version of the AMOS Club/Shuffle extension, allows you to

read the keyboard directly, even when Forbid is active. Nigel Critten's Sticks extension lets you read the mouse (in either port) when Forbidden also.

To use Forbid, there are no parameters, so it's just

```
Z=Execall(-132)
```

and to Permit

```
Z=Execall(-138)
```

Basically, if you have Shuffle and the Stick extensions or are just using Joystick control in your game, stuff a Forbid at the start and a Permit at the end to get a little extra speed !

Also for extra speed when just calculating (although it isn't much more speed), you could try using Disable and Enable, which Disable and Enable all the interrupts. If you use Disable, none of the interrupt based routines will work (Bobs, Sprites, Flash, Screens, in fact everything that makes AMOS worthwhile !) but everything else (i.e. numbercrunching) will be slightly faster. Unfortunately I can't get an accurate timing of the difference in speed (The Timer function is also interrupt based !), but it seems to be a small speed increase. Thus, if you set your Mk1 AMOS Universe and Utterly everything analyser going with Disable set, it might take 9,999 years to complete instead of 10,000 !

To use Disable do:

```
Z=Execall(-120)
```

and to enable again do:

```
Z=Execall(-126)
```

Most of the system functions have been used in the various extensions now, so you don't actually have to use them in their raw form. However, it is always useful to know how it works !

Converting from AMOS to Easy AMOS

If you are learning to program, then Easy AMOS is a great introduction to the language. However, rather a lot of the programs written for AMOS, to demonstrate various programming tricks, including a lot in the newsletter, use commands that Easy AMOS doesn't have. Because of which, you will need to do some converting before using the program.

Basics

The first thing you need to do, is to get the program into Easy AMOS itself. EA will actually let you enter commands that are in AMOS (except for **Get Sprite Palette**, which you have to enter as **Get Bob Palette**). You then attempt to run the program, and you will more than likely get the error :

Instruction not available in Easy-

AMOS at line xxxx

The line will be listed, with arrows pointing at the command that you will have to convert. Unfortunately, some of the more basic commands in AMOS were not included, so there will be quite a few cases of this. However, most (apart from AMAL) will be able to be converted to an equivalent bunch of Easy AMOS commands !

Your first job, upon finding that EA won't run the program, is to check to see if AMAL is used. If so, try another program, as Easy AMOS doesn't support AMAL in any form. Sprites cannot be used either, so any programs using Sprite commands will probably require major conversion.

If AMAL or sprites aren't used, go

through each line in the program, and break it into separate commands, by using the Return key wherever a colon (:) is used. If you convert the Decoloniser program from Volume 1 Issue 5/6 (or if you have AMOS, run it in AMOS), it will automatically separate the program into 1 command per line for you (although you will have to save it as ASCII first, and then merge it back in again once converted).

If you separate the commands out, even the highly condensed 10-liner programs become a lot more readable and easy to convert.

Now comes the fun bit, searching out the AMOS only commands, and replacing them with EA equivalents. This is where the Find function (Alt F3 for the first one and Ctrl - N for the next) comes in handy. The commands in **Table 1** (Over the page) are really easy to change, simply select the file menu and click on Replace All (Alt F5). Hit the 'Y' key, or press on the 'Yes' with the mouse button and enter Get Sprite Palette as the string to search. You will then be asked for the string to replace it with, enter Get Bob Palette. Press Return, and EA will automatically change the Get Sprite Palette into Get Bob Palette. If you loaded the program as a .AMOS file, then you won't need to do this for Get Sprite Palette, as EA will do this automatically. However, the commands in Table 1 can be changed with Replace All.

Other commands you will have to manually convert, but they are easy enough. For instance, the **Inc N** and **Dec N** commands. These convert to **N=N+1** and **N=N-1** respectively. The

Add command is simple enough if it is just **Add N,X** which becomes **N=N+X** but if it is the Add command with the extra 2 parameters to specify the range to keep the variable in, you have to be sneaky. For instance,

Add A,1,0 To 7

would become

A=(A+1 Mod 8)

if the first value isn't a zero or if the amount being added is a variable, or is negative, such as

Add A,1,-20 to 40

you will have to do the following:

Do the command **A=A+ the second**

parameter i.e **A=A+1**

Then do: **If A<third parameter**

:A=A+Fourth Parameter-Third

Parameter+1 : Endif i.e

If A<-20 : A=A+40-20 : Endif (The

-20 can be shortened to +20 as any good mathematician knows !)

and then do: **If A>fourth**

parameter : A=A-Fourth

Parameter+First Parameter-1 :

Endif i.e

If A<40 : A=A-40+ -20 : Endif

(Again, the + -20 shortens to -20)

This will cope with the Add command, although making it longer!

To find the Add, Inc and Dec commands, use the Find function again !

Swap also becomes tricky. For this one, you need to revert to the old system of having a third variable.

For instance, **Swap A,B** would

become **C=A : A=B : B=C !**

Well, hopefully this will give you a lot more programs to play about with.

Have fun !

Table 1. Commands that can be converted with Replace All

AMOS (search)	EA Equivalent (Replace)
Screen Height	Deek(Screen Base+76)
Screen Width	Deek(Screen Base+78)
Screen Colour	2^Deek(Screen Base+80)
Space\$(	String\$(" " (one space is between the quotes)
Jleft(1)	-Sgn(4 and Joy(1))
Jright(1)	-Sgn(8 and Joy(1))
Jup(1)	-(1 and Joy(1))
Jdown(1)	-Sgn(2 and Joy(1))

If the program is a 2 player game, where a variable is used to state the stick number, then you will have to manually convert the joystick commands, replacing the 1 in the brackets with the variable. e.g =Jleft(N) would become =-Sgn(4 and Joy(N)).

Normal AMOS Users !

Just because you aren't using Easy AMOS doesn't excuse you from reading this article ! Going on some of the mistakes a lot of the more experienced programmers have been

making with AMOS, it sounds like some of you have become too used to the "Super" commands in AMOS. So I would suggest going back to doing some of your programs the "Old Fashioned" way, just for practise !

Hardware Scrolling using AMOS TOME

Hardware scrolling is something that AMOS programmers have been attempting to do properly since AMOS was first written. Unfortunately, due to some inconsistencies in the Screen Offset command and the low speed that AMOS handles screen drawing, it hasn't really been practical. Until now that is, the TOME system updates the screen fast enough and by using the Screen Offset command in a more limited way and taking advantage of faster ways of using the Screen Copy command we can just get away with it. Once TOME 4.5 is generally available (i.e when I've finished it !), its double speed map plotting and ultra high speed screen copy (written specifically for this type of routine) make the scrolling even better.

The following example will scroll a map in all directions, using Screen Offset to handle the horizontal scrolling (which it is good at) and Screen Copy to handle the vertical (which Screen Copy handles better). By using TOME and a fractionally larger screen than normally used for a TOME map display you can now hardware scroll around huge areas, without running out of memory !

This program will run reasonably smoothly interpreted, without any special commands, although for best results it should be compiled, with the Forbid & Unforbid Execall functions used (Z=Execall(-132) and Z=Execall(-138)). It is written using TOME Series IV, although you can use TOME 3 (see

the rems for where to modify) although the scrolling won't be as good.

```
1) Dir$="TOME4:" [90]
2) Load "TILES/Mazemantiles.abk" [63]
3) Reserve As Work 6,2504 [125]
4) Load "Maps/mazeman_map.abk",6 [41]
5) Load "tiles/mazeman_sprites.abk" [92]
6) Tile Size 16,16 [61]
7) Map View 0,0 To 336,208 :
Rem set to screen area+1
tile in each scroll
direction [43]
8) Screen Open 0,336,192,32,
Lowres : Curs Off : Flash
Off : Cls 0 [59]
9) Screen Open 1,336,208,32,
Lowres : Curs Off : Flash
Off : Cls 0 : Screen Hide 1
[77]
```

This is for a 320x192 screen display

```
10) Screen 0 [20]
11) Get Icon Palette [33]
12) Wait Vbl [82]
13) Screen Display 0,128,,
320, [119]
14) Double Buffer [26]
15) No Icon Mask [63]
16) Map Handle Init : Rem
hxo=-1 for TOME3 [41]
17) MXX=Map X*16-320 [117]
18) MXY=Map Y*16-192 [126]
19) MX=MXX/2 : MY=MXY/2 [58]
These hold the maximum MX and MY values for the loaded map
20) Bob Update Off : Auto
View Off [70]
21) X=160 : Y=100 [60]
22) S=2 : Rem S is the scroll
speed [70]
Adjust for best results
23) While Fire(1)=0 [90]
```

```

24) HX=Map Hx(MX) : Rem hx=mx
/16 for TOME3 [94]
25) FX=MX and $F [19]
26) HY=Map Hy(MY) : Rem hy=
my/16 for TOME3 [95]
27) FY=MY and $F [19]
28) Rem add in HXO=HX : HYO=
HY for TOME3 [92]
29) Screen 1 [21]
30) Map Handle 1,HX,HY : Rem
gosub nicemap for TOME3 [92]
31) Screen 0 [20]
32) Bob 1,X+FX,Y,1 : Rem put
your bobs here [89]
33) Screen Copy 1,0,FY,336,
192+FY To Logic(0),0,0 : Rem
copy the entire width [49]
34) Bob Draw [78]
35) Wait Vbl [82]
36) Screen Swap [21]
37) Screen Offset 0,FX,0 :
Rem only use screen offset
for X [50]
38) View [44]
39) Wait Vbl [82]
40) Rem now control the
character [74]
41) X=X+(Jleft(1)-Jright(1
))*S [3]
42) Y=Y+(Jup(1)-Jdown(1))*S
[84]
This is just a simple move routine,
you can easily modify to your
favourite.
43) Rem push scroll effect
[94]
44) If X<48 and MX>0 : MX=MX-
S : X=X+S : End If [71]
45) If X>272 and MX<MXX :
MX=MX+S : X=X-S : End If [45]
46) If Y<48 and MY>0 : MY=MY-
S : Y=Y+S : End If [71]
47) If Y>144 and MY<MYX : MY=
MY+S : Y=Y-S : End If [42]
48) X=Max(8,Min(312,X)) [43]
12

```

```

49) Y=Max(8,Min(184,Y)) [38]
50) Wend [57]
As soon as I finish off TOME 4.5 (mid
June 93 at the earliest) I will be
releasing it as a P.D updater disk, so
anyone with TOME Series IV can
update at minimal cost. Other bits to
be modified will be an even better
animation system and even better
Shuffle Extension. I don't mind you
calling up to ask "is it available yet?"
on this one, as you can point out any
bits you would like adding or modifying
while you're at it!

```

The above routine is being used for the games in the TOME Goodies Disk 2, and space allowing, the TOME 4.5 updater will also be on this disk. (You will need TOME IV for Goodies 2).

TOME Editor Fixes

A Few minor bugs have managed to sneak into the TOME Series IV Editor (TOME.AMOS). Nothing life threatening, just sometimes a little irritating!

Line 346 : If using AMOS 1.33 or earlier, change to:
Play Off

Although getting AMOS 1.34 is a better idea!

Line 1277: Change to:
If Z=2 : ANM=Max(0,ANM-1+(K=2)*9)

Line 1280: Change to
If Z=3 : ANM=Min(NANIMS-1,ANM+1-(K=2)*9)

These allow you to jump 10 anims at a time when selecting the animation to edit.

Line 1307: Change to
Z=Z-22+MATL : A=ANT

To allow you to use all 44 anims with 32x32 pixel tiles.

Once you have done this, save it to your TOME Backup disk.

Using the Float Bob Commands

One of the strangest looking commands added in the later versions of the Shuffle Extension (V2 upwards) is the Float Bob command. It is intended to be used in games where a "floating" number of bobs need to be available at once. I.e where you can't assign bob numbers to objects on screen, because you never know how many will be on screen at once.

In most games (especially those that have scrolling or changing backgrounds) this system is much faster.

How it works

The standard floating bob system would normally use a variable such as NB (Next Bob) as an offset for the Bob number. You would put the bob on the screen with **Bob 1+NB,X,Y,I** and then add 1 to NB. At the end of the frame, NB would be compared with what it was in the previous frame, and if it is less, then all the bobs that were put on the screen in the previous frame, but are no longer used are removed.

This system works fine, but the Float Bob commands speed it up by having their own internal counter instead of using NB. So you do the following sequence:

```

Float Bob Reset : Rem the
same as NB=0
Float Bob 1,X,Y,Z
Float Bob 1,X2,Y2,Z2
....etc....
Screen Copy 1 to logic(0) :
Rem copy background to
logical screen
Float Bob Clear : Rem this
removes any unused bobs
Bob Draw

```

Screen Swap

Notice that the Bob number used is always the same. This is because the internal counter is being used as an offset to the actual bob number. So the first bob put on the screen will be bob number 1, the second bob number 2 and so on.

If at any point you need to find out what bob number was used, the =Last Float Bob function returns the bob number used for the last float bob put on the screen.

Collision Detection

Because the bob numbers vary for the images on screen, you can't do a normal =Bob Col() collision detection. Instead, you have to Dimension an array to the maximum number of bobs you will have on screen at once, and use the =Last Float Bob function to record either the image number or information about the bob. For instance:

```

Float Bob 1,X,Y,I : Rem Plyer
B(Last Float Bob)=0
PB=Last Float Bob
Float Bob 1,EX,EY,ENEMY
B(Last Float Bob)=2:Rem Enemy
Float Bob 1,BX,BY,BULLET
B(Last Float Bob)=1:Rem Shot
This way, you can easily do BC=Bob
Col(PB) to check for the collisions.
Then checking the Col() array, you can
tell what you hit:
For A=PB+1 To Last Float Bob
If Col(A) and B(A)=1
Rem Hit an enemy
Endif
Next A

```

In future versions of the Shuffle Extension, this system will be automated somewhat, and commands such as =First Col(A To B) and =Nxt Col will make the collision detection much easier and faster.

10 Liners

By Aaron Fothergill

New AMOS users seem to shy away from the fun of 10 liners, as they often look quite intimidating. So the first 10 liner this issue is a splitter program, that will separate out a 10 liner into a single statement per line program, which is a lot easier to read. Once you have typed in and saved the program, press Ctrl-A (to mark it as a block), save it as ASCII (from the Block menu) and use it to split itself! When you want to see how a 10 liner is made up, simply save it as an ASCII file, run the 10 line splitter, clear the program and MERGE the resulting split up program back into AMOS. While I was at it, I wrote a 10 liner writer by the way, which does the reverse and crams a normal program into as few lines as possible. While not as efficient as hand coding 10 liners, it's still quite effective! The 10 Liner Maker is on Issue 4 of The Disk, along with all the other programs from this issue.

```
1) ' 10-liner splitter upper
[37]
2) Dim S$(500),L$(500) : Set
Input 10,-1 : S=0 : F$=Fsel$(
"*.ASC","","Load an ASCII
Listing") [64]
3) If F$<>"": If Exist(F$) :
Open In 1,F$ : While Eof(1
)=0 : Line Input #1,S$(S) :
Inc S : Wend : Close 1 [35]
4) A=0 : While A<S : P=1 :
L$="" : Q=0 : LC=0 : While
P<=Len(S$(A)) [85]
5) P$=Mid$(S$(A),P,1) : If
(P$=":") and Q=0 : L$(L)=L$
: L$="" : P$="" : Inc L :
LC=P+1 : End If [87]
```

```
6) If P$=Chr$(34) : Q=1-Q :
End If [92]
7) If Mid$(S$(A),P,4)="Then"
and Q=0 : L$(L)=Mid$(S$(A),
LC) : P$="" : L$="" : P=999
: Inc L : End If [14]
8) L$=L$+P$ : Inc P : Wend :
If L$<>"": L$(L)=L$ : Inc L
: End If [83]
9) Inc A : Wend : F$=Fsel$(
"*.asc","","Save an ASCII
Listing") : If F$<>"": [70]
10) Open Out 1 , F$ : For
A=0 To L-1 : Print #1,L$(A)
:Chr$(10); : Next A : Close
1 [46]
11) End If : End If : End If
[65]
```

Now for another in our series of "Classic 8 bit games that can now be done in 10 lines!". Do you remember Robotron, or more recently Llamatron, where you start in the middle of the screen, surrounded by hordes of baddies, all out to get you, when you just happen across a laser blaster. The rest (and the baddies) is history! Well XoTron (a bad pun, I know!) is the basic game in a mere 8 lines! You will need the following graphics though.

a) A packed screen (we used a 32 colour one for testing) packed into bank 5.

b) A few sprites:

1-8 are the player, rotated in 8 directions clockwise from facing left, left & up, up, right and up etc.

9-16 are the laser bolt images, rotated to match the player through 8 directions.

17 upwards are the baddies. 2 images for each (so they animate). Colours 8-11 cycle (using the Shift Up command at the end of line 3). The game is set up to use as many baddies as you want to draw, so do a few!

Once you have your graphics (remember the version of XoTron on Issue 4 of The Disk has all the graphics done), type in the following program, save it and start blasting. This one is even better compiled!

```
1)' Xotron (C) Aaron Fothergill
Shadow Software 1993 [33]
2)' Written for the AMOS Club
[74]
3) H=256+Ntsc*56 : X=160 :
Y=H/2 : Unpack 5 To 0 : N=(
Length(1)-17)/2-1 : Dim X(20
),Y(20),S(20),DX(7),DY(7), BX
(9),BY(9),FR(2,2),R(9) : Hide
: Screen Display 0,,50+ Ntsc
*5,,H : Shift Up 4,8,11,1[69]
4) A$="123004765" : For A=0
To 2 : For B=0 To 2 : FR(B,A)
=Asc(Mid$(A$,A*3+B+1,1))-48 :
Next B : Next A : For A=0 To
1 : S(A)=1 : Next A : A$="010
0102021221202" : For A=0 To 7
: DX(A)=Asc(Mid$(A$,A*2+1,1
))-49 [182]
5) DY(A)=Asc(Mid$(A$,A*2+2,1
))-49 : Next A : Get Sprite
Palette : Double Buffer : L=1
: C=-2 : While Bob Col(1,20
To 40)=0 : Bob 1,X,Y,1+FR(1+
JL-JR,1+JU-JD) : If B : For
A=0 To B-1 : Bob 2+A,BX(A),BY
(A),R(A)+9 [216]
6) Next A : End If : For A=0
To L : Bob 20+A,X(A),Y(A),S
(A)+T : Next A : Wait Vbl :
For A=0 To L : Add X(A),Sgn(X
-X(A)) : Add Y(A),Sgn(Y-Y(A))
: If Bob Col(A+20,2 To 11) or
S(A)=1 : S(A)=Rnd(N)*2+17 :
S=Rnd(1) [197]
7) W=1-S : X(A)=Rnd(319)*S :
Y(A)=Rnd(H)*W : X(A)=-X(A)
=0)*Rnd(1)*319+X(A) : Y(A)=-
(Y(A)=0)*Rnd(1)*H+Y(A) [33]
```

```
8) Inc C : End If : Next A :
JU=Jup(1) : JD=Jdown(1) :
JL=Jleft(1) : JR=Jright(1) :
Add X,(JL-JR)*2 : Add Y,(JU-
JD)*2 : If Fire(1) and B<4 :
BX(B)=X : BY(B)=Y : R(B)=
FR(1+JL-JR,1+JU-JD) [199]
9) Inc B : End If : If B :
For A=B-1 To 0 Step -1 : Add
BX(A),DX(R(A))*8 : Add BY(A)
,DY(R(A))*8 : If BX(A)<-16 or
BX(A)>336 or BY(A)<-16 or BY
(A)>H+16 or Bob Col(A+2,20 To
40) : Swap BX(A),BX(B-1) [80]
10) Swap BY(A),BY(B-1) : Swap
R(A),R(B-1) : Bob Off B+1 :
Dec B : End If : Next A : End
If : If C>2*(2^L) : L=Min(20,
L+1) : S(L)=1 : End If : T=1-
T : Wend : Paper 0 : Home :
Centre "You Got"+Str$(Max(C,
0)) : Wait Key [45]
```

Robotron, and Defender were both suggested to me by Phil South as games to ten linerfy (I think jokingly), so here's TreeFender. Stop your forest from being stolen by yellow JCB alien dudes. It's hardware scrolling, fast and fun. Again, you'll need some graphics for this one, but they're basically just a few bobs:

- 1: 8x8 pixel tree, hot spot at base
- 2: Alien ship 16x16. Hotspot at centre
- 3: Your ship, facing left & 4: Your ship facing right. Hotspot centre, and 16x16 pixels
- 5: Laser bolt, 16x2 solid block. Hotspot at right.
- 6: 8x200 pixel vertical line, using only 2 bitplanes, the following will grab it for you:

```
Screen Open 0,320,200,4, Lowres
curs Off:flash off:cls 3
Get Bob 6,0,0 to 8,200
Obviously, only do this after you have
```

created and loaded the other 5!

```
1) Screen Open 0,960,200,16,0
:Flash Off : Cls 0:Get
Sprite Palette : Dim Y(59),
TX(9),TY (9),DX(5),DY(5),DD
(5),AX(5), AY(5),T(9),Y1(9)
: Auto View Off : Bob Update
Off : For A=0 To 3 : AX(A)=
Rnd(959) [56]
```

```
2) AY(A)=16 : DD(A)=1-Rnd(1)*
2 : DY(A)=Rnd(1)+1 : Next A
: Y=200-Rnd(32) : YO=Y :
Hide : For X=15 To 959 Step
16 : Ink 12+(Y>190)+(Y<160)
*11 : Draw X-16,YO To X,Y :
YO=Y : Y=Max (99,Min(199,Y+
Rnd(8)-Rnd(8))) [61]
```

```
3) Y(X/16)=YO:Next X: Double
Buffer : Screen Display 0,
128,,320, : For A=0 To 9 :
TX(A)=16*(Rnd(58)+1): TY(A)
=Y(TX(A)/16) : Bob 3+A,TX(A)
,TY(A),1 : Y1(A)=TY(A) :
Next A : X=320 : Y=100 :
L=10 : N=L [200]
```

```
4) While L>0 and D<20 : Bob
Clear : Bob 1,X,Y,3-(LD>0) :
For A=0 To 3 : Bob 32+A,AX
(A),AY(A),2 : B=T(A)-3 : If
B>-3 : Bob B+3,TX(B),TY(B),
: TX(B)=AX(A) : TY(B)=AY
(A)+6 : End If : Next A : If
T mod 2=0 [170]
```

```
5) For A=0 To 9 : TY(A)=Min
(Y1(A),TY(A)+1) : Bob 3+A,,
TY(A), : Next A : End If :
Bob 2,BX,BY,5 : Bob 0,Max(
0,X-160),D*10,6 : Bob Draw :
Wait Vbl : Screen Swap :
Screen Offset 0,Max(0,Min
(639,X-160)), [200]
```

```
6) View : Z=(Jleft(1)-Jright
(1)) : DX=Max(-12,Min(12,DX+
Z*2+Sgn(DX)*Q)) : X=Max(0,
Min(959,X+DX)) : If Fire(1)
```

16

```
and BX=0 : BX=X : BY=Y :
BD=LD*16 : R=12 : End If :
Y=Max(0,Min (199,Y+DY)):SD=
Sgn(DY)*Q [200]
7) DY=Max(-8,Min(8,DY+(Jup(1)
)-Jdown(1))+SD)) : Add D,-
Bob Col(1,32 To 37) : If Z :
LD= Sgn(Z) : End If : For
A=0 To 3 : If Bob Col(A+32)
and DY(A)>0 and T(A)=0 :
For B=0 To 9 : If Col(B+3)
and TY(B)>0 [216]
8) T(A)=B+3 : DY(A)=-1 : B=99
: End If : Next B : End If :
Add AX(A),DX(A) : Add DX(A),
DD(A) : If Abs(DX(A))=8 :
DD(A)=-DD(A) : End If : Add
AY(A),DY(A) : If AY(A)>190
: DY(A)=-1 : End If : If
AY(A)<-7 [210]
9) If T(A)>0 and DY(A)<0 :
Y1(T(A)-3)=-8 : T(A)=0 : Dec
L : Dec N : End If : DY(A)=1
+Rnd(1) : AX(A)=TX(Rnd(9)) :
End If : Next A : Add BX,BD
: Dec R : BX=-BX*(R>0) : If
Bob Col(2) : For A=0 To 3 :
If Col(A+32) [216]
10) T(A)=0 : Inc SC : DY(A)=-
1 : AY(A)=-8 : End If : Next
A : End If : Add T,1,0 To 3
: Q=(T=0) : Wend : Screen
Offset 0,320, : Paper 0 :
Centre "Score"+Str$(SC*100
+N*250) : Update : View :
Wait Key [64]
```

Other 10 liners to go on this issue (4) of The Disk are the 10 line 3D maze game (flips TOME Maps up into 3D so that you can walk around them!), Joist (A Joust variant) and Super Squint, a 10 line car racing game for 1 or 2 players!

Foreign Affair

By Aaron Fothergill

Writing Multi Lingual Programs

One thing I noticed when at the OZ Happening 92 in Belgium, was that a lot of the programs I had written previously in AMOS had problems with the French keyboards we were using there. This was because I was using a U.K copy of AMOS (with the U.K keymap) on a French keyboard, but also because I was using the Key State() function, which literally regards the keyboard as a set of buttons rather than characters (This is why it is normally better for games than the buffered Inkey\$ command). I have also had a lot of queries from programmers who are writing multi-lingual games (usually on CD), who need to make the program switch languages for their prompts and text.

Fortunately, with a little forethought, you can quite easily make your program multi-lingual, and able to redefine its keyboard useage. It's all a matter of arrays!

Strings

If your program requires text strings for prompts etc. store them all in an array. This array can either be loaded up from disk as a sequential file, or as a bank and using the =Bank Str\$() functions from Shuffle. If you have plenty of memory, or few strings, and want to make the language switch instant, you can use a multi-dim'med array, with the prompts for each language in a different level.

Rem Single Language at once

Dim P\$(5),L\$(2)

L\$(1)="Prompts_English"

L\$(2)="Prompts_Francais"

Input "Select (1) English or

(2) Francais";EF

Open In 1,L\$(EF)

For A=1 To 5

Line Input #1,P\$(A)

Next A

Close 1

Print P\$(1)

The above will load in 5 prompts in either English or French (I'm hopeless at French, so sorry no examples!), which you would save to the disk as Prompts_English for the English ones and Prompts_Francais for the French ones. Thus if the above program needed to ask "Do you speak English (Y/N)?" as part of the program where the **Print P\$(1)** is, the first prompt in Prompts_English would be:

Do You Speak English (Y/N) ?

and in Prompts_Francais it would be:

Parlez Vous Anglais (O/N) ?

Notice the O/N instead of Y/N (for Oui/Non instead of Yes/No) it's strange how little bits like that can catch you out! Basically, any strings you use in your program would need to be pulled from the Prompts file and stored in the P\$() array.

The Keyboard

If, like me, you tend to use the **Key State()** or **Key Scan** functions to read the keyboard, you can never know what letter will be on the key that you are looking at. Also, you might want the keyboard to be re-definable in your game. When keyboard control was popular for games, there were two main camps in the "What keys" argument, either A/Z and </> for left/right or I/K with J/L for left/right. Basically, you just set up a numeric array with all the key numbers for the keys you want to use as a default in it. So K(1) would be your first key, say A (32), K(2) would be the Key State number for Z (49), with 3

& 4 the Key State codes for < & > (56 & 57). When you want to redefine the keyboard, you simply change the values in this array. The following example shows how you could use this system:

```
Dim K(4) : K(1)=32 : K(2)=49
K(3)=56 : K(4)=57
Do
If Key State(K(1)) Then Print
"Up"
If Key State(K(2)) Then Print
"Down"
If Key State(K(3)) Then Print
"Left"
If Key State(K(4)) Then Print
"Right"
```

Loop

To redefine the keyboard, just change the values in K(1) to K(4). The following program can be used to get the Key State code for any key:

```
Do : For A=0 To 127
If Key State(A) Then Print A
Next A : Loop
```

Smaller Screens

Again, with a little planning, you can make your program automatically re-size to an NTSC screen. For instance when opening a screen, instead of:

```
Screen Open 0,320,256,16,Lowres
do:
```

```
Screen Open 0,320,256+Ntsc*56
,16,Lowres
```

The Ntsc variable is set to 0 (False) if you program is running on a PAL screen, and -1 (True) when running on an NTSC screen. So whenever drawing anything to the screen, you can use it to calculate any vertical co-ordinates. For example

Box 0,0 To 319,255 would become:

```
Box 0,0 To 319,255+Ntsc*56
```

Go on, make an American happy, write an NTSC compatible game today!

Disk Lister and File Finder

by Aaron Fothergill

I have been asked by several AMOS Programmers how to get the filenames of all the files on a disk, so that they can be catalogued, checked for viruses/damage etc. or so that particular files can then be searched for and modified or viewed. The following program will scan an entire disk (including hard disks), recording all the filenames and directory names, and then save them to disk as a catalogue. You can modify the program in any way you want. One suggested modification is the Finder, listed after the main listing. If you replace the Save routine (from the rem to the End statement) with the finder routine, it will search for any file on the disk.

The program works by listing all the files in the current directory, with the **Dir First\$** and **Dir Next\$** commands, and then scanning through the list of files and directories that results for any directories (marked by an asterisk in from of them by the Dir First/Next commands). If it finds a directory in the list, it will scan through that directory before going back to looking at the list for the next directory. This will continue until it cannot find any more files.

The same sort of routine is used in SpriteX 3.0's file selector (so that you can search for files) and in the latest TOME IV & Shuffle Installation programs (So that your AMOS_System directory can be automatically found). It comes in useful for all sorts of routines, especially Link Virus Killers, which need to check all the files on a

disk.

This particular version is able to scan a 15-20 megabyte hard disk partition. For larger scans, increase the **Set Buffer** command in line 1, and the **Dim F\$()** and **D\$()** statements in line 4 if you think you have more than 5000 files.

```
1) Set Buffer 150 [121]
2) Screen Open 0,640,256+
Ntsc*56,2,Hires [85]
3) Curs Off : Flash Off : Cls
0 : Paper 0 : Pen 1 : Colour
1,$FFF [85]
4) Dim F$(5000),D$(5000) [79]
5) N=0 [67]
6) F=1 [74]
7) D$="" [92]
8) Rem throw in a dir$="
whatever:" here to select a
drive [60]
9) While F<>0 [0]
10) D2$=D$ [8]
11) If D2$<>" " : D2$=D2$+"/"
: End If [37]
12) Print "Searching
Directory ";D$ [45]
13) F=0 [75]
14) F$=Dir First$(D2$+"*")
[38]
15) While F$<>" " [19]
16) Gosub NOSPACES [117]
17) F$=Dir Next$ [15]
18) Wend [57]
19) For A=0 To N-1 [113]
20) If Left$(F$(A),1)="*" [56]
21) F=1 [74]
22) D$=D$(A)+Mid$(F$(A),2)
[24]
23) F$(A)=" "+Mid$(F$(A),2)
[77]
24) A=99999 [69]
25) End If [64]
26) Next A [97]
27) Wend [57]
```

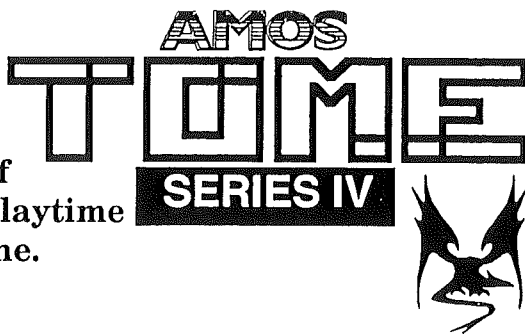
```
28)' save it all to disk [85]
29)' first get the filename
[82]
30) F$=Fsel$("*.*CAT","", "Save
Disk Catalogue as ?") [78]
31) If F$<>" " [78]
32) Print "Saving" [122]
33) Open Out 1,F$ [36]
34) For A=0 To N-1 [113]
35) Print #1,D$(A); " ";F$(A)
;Chr$(10); [51]
36) Next A [97]
37) Close 1 [71]
38) End If [64]
39) End [107]
40) NOSPACES: [45]
41) I=Instr(F$," ",2) [86]
42) F$(N)=Mid$(F$,1,I-1) [103]
43) D$(N)=D$ : If D$<>" " : D$
(N)=D$(N)+"/" : End If [76]
44) Inc N [43]
45) Return [42]
```

This is the Finder routine. If you use it to replaces lines 28-38 in the above listing, it will allow you to find where a file is on disk.

```
1) ' Find a file variant [51]
2) Print "Enter Filename to
find" [25]
3) Line Input F$ [59]
4) For A=0 To N-1 [113]
5) If Instr(Upper$(F$(A)),
Upper$(F$)) [51]
6) Print "Found. at directory
";D$(A); " ";F$(A) [77]
7) End If [64]
8) Next A [97]
```

If you have any more requests for routines that might be useful to other AMOS users, send them in! Unfortunately, due to the number of enquiries I get, I can only rarely reply directly to any query by post, but it may get into the Newsletter!

"AMOS TOME has everything in it, I'll never use any other map system", Len Tucker, Programmer and author of AMOS in Action, Noddy's Playtime and Totally AMOS magazine.



Well, Len Tucker thinks its the bee's knees !
What are you going to say about AMOS TOME ?

"Go and buy it now.." ●●●●○ : Phil South, Amiga Shopper magazine

"I can heartily recommend the AMOS TOME system to any serious AMOS user" : Stephen Hill, AMOS Game Maker's Manual

"TOME.. has now established itself as the extension system no AMOS programmer can afford to be without." 88% :Nick Veitch, CU Amiga Magazine

AMOS TOME is the ultimate extension for AMOS. TOME Series IV comes with 67 commands, TOME Editor (the most powerful available on any computer), 4 demo games (all sourcecode), 80 page manual and example programs for all the commands. AMOS TOME works with AMOS 1.34, 1.36 and the Compiler and with all Amigas (including the 1200 and 4000).

Please Send me:

AMOS TOME Series IV

☐ £24.99

AMOS CText V2.5

☐ £10 (£12 Overseas)

The Disk Issue 4

☐ £2.50 (£3.00 Overseas)

A Subscription to The Disk

(includes Back Issues)

☐ £7.00 (£10 Overseas)

Name Membership #

Address

..... Postcode

Make cheques (in pounds sterling only please) payable to "Shadow Software" and send orders to Shadow Software, 1 Lower Moor, Whiddon Valley, Barnstaple, N.Devon. EX32 8NW. England